60000286

CONTROL DATA

CYBER 96X MSL MAINTENANCE SOFTWARE REFERENCE MANUAL

MAT3/P TEST DESCRIPTION

CDC® COMPUTER SYSTEMS
CYBER 960, 962

PREFACE

This manual describes the CDC CYBER 960 and 962 Computer System's MAT3/P test. Procedures for using this test are provided in the CYBER 96X Test Procedures Reference manual.

NOTE

It is assumed and planned that the user of program descriptions will have an in-depth understanding of the processor logic; including familiarity with all levels of the Multi-Level Block Diagrams.

DISCLAIMER

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or undefined parameters.

REVISION RECORD

REVISION	DESCRIPTION
A (07-11-88)	Manual released.

Publication No.
60000286

REVISION LETTERS I, O, Q, S, X, AND Z ARE NOT USED

1988

by Control Data Corporation
Printed in the United States of America

Address comments concerning this manual to:

Control Data Corporation
Technology and Graphics Division
4201 Lexington Avenue North
St. Paul, Minnesota 55126-6198

or use Comment Sheet in the back of this manual.

CONTENTS

MAT3. MEMORY ARRAY IDENTIFICATION TEST..... 1

General..... 1
Section Descriptions..... 6
    Section 0 - Test Data Retention Ability with Suppression of Refresh-related Faults..... 6
        Section 0, Subsection 0..... 6
    Section 1 - Test Data Retention Ability without Suppression of Refresh-related Faults..... 10
        Section 1, Subsection 0..... 10
    Section 2 - Test Address Lines by Testing for Address Uniqueness..... 13
        Section 2, Subsection 0..... 13
    Section 3 - Test with Multiple Accesses to Each Address, with Random Data..... 18
        Section 3, Subsection 0..... 18
    Section 4 - Test with Random Data..... 21
        Section 4, Subsection 0..... 21
    Section 5 - Test All of Testable Memory with Random Data and Random Addresses..... 27
        Section 5, Subsection 0..... 27

APPENDIXES

A. GLOSSARY..... A-1

B. MEMORY MAPS..... B-1

THIS PAGE HAS INTENTIONALLY BEEN LEFT BLANK

MEMORY ARRAY IDENTIFICATION TEST, MAT3/P

==========

GENERAL

MAT3/P tests the memory array boards in CYBER 960 and 962 Computer Systems. It assumes that all memory logic external to the memory array boards has been previously tested. It also assumes that the processor and IOU have been fully tested. To decrease execution time, the actual reading and writing of memory are done by microcode in the CPU. Failure data, when analyzed, is intended to aid in failure isolation.

All CPU activity (micrand execution) is controlled by a previously tested PPU. Controlling signals are sent to and received from the CPU via the Maintenance Channel and the Maintenance Access Control unit (MAC).

A typical test sequence is:

1. Load the CPU registers with desired data.
2. Send the CPU a "Start" command.
3. The CPU executes the diagnostic micrand sequence and halts.
4. The PPU senses halt and directs the MAC, via PPU-resident code, to read test results from the CPU and memory registers.
5. The data retrieved is examined for failure.

All sections, subsections, and conditions should be executed in sequential order. Successful completion is predicated on the successful completion of previously executed sections, subsections, and conditions.

MAT3/P TEST STATUS FORMAT:

Certain processor status register bits are grouped and formatted into a 32-bit word, which is reported in the right half of the "Actual Results" for those test conditions that execute a test micrand sequence. For this reason, the 32 bits in the Status Word have been numbered 32 through 63.

Bits 32 through 47 of the Status Word are taken from the memory's Status Summary register and the Processor Fault Status registers after execution of the test micrand sequence. Consult the chart below for their format and meaning.

Bits 48 through 63 of the Status Word describe the "halt status" of the micrand sequence that executed. This field details whether or not the micrand sequence set the "Processor Halted" bit in the Status Summary register in the time allotted to it by MAT3/P and, if so, at what Control Store address it halted. Consult the chart below for the format and meaning of these bits.

Status Word Bit Number	Contents	
32	PFS Reg. 84, bit 48;	DAI data parity error, byte 0
33	PFS Reg. 84, bit 49;	DAI data parity error, byte 1
34	PFS Reg. 84, bit 50;	DAI data parity error, byte 2
35	PFS Reg. 84, bit 51;	DAI data parity error, byte 3
36	PFS Reg. 84, bit 52;	DAI data parity error, byte 4
37	PFS Reg. 84, bit 53;	DAI data parity error, byte 5
38	PFS Reg. 84, bit 54;	DAI data parity error, byte 6
39	PFS Reg. 84, bit 55;	DAI data parity error, byte 7
40	PFS Reg. 80, bit 0;	Uncorrected error
41	(Not used)	
42	PFS Reg. 82, bit 58;	CM uncorrected write
43	PFS Reg. 82, bit 59;	CM uncorrected read
44	PFS Reg. 82, bit 60;	CM reject
45	Mem SS bit 61;	CM uncorrected error
46	PFS Reg. 84, bit 42;	Deadman time-out
47	(Not used)	
48	(Program Flag);	Processor did not halt.
49	(Not used)	
50	(Not used)	
51	Not shadow memory *	
52	Not extended control store *	
53	Micrand Address Reg. bit 0 *	
54	Micrand Address Reg. bit 1 *	
55	Micrand Address Reg. bit 2 *	
56	Micrand Address Reg. bit 3 *	
57	Micrand Address Reg. bit 4 *	
58	Micrand Address Reg. bit 5 *	
59	Micrand Address Reg. bit 6 *	
60	Micrand Address Reg. bit 7 *	
61	Micrand Address Reg. bit 8 *	
62	Micrand Address Reg. bit 9 *	
63	Micrand Address Reg. bit 10*	

* These bits are the contents of the Micrand Address register, as read after the processor has halted (or timed out). The normal address is 3. For the meaning of other halt addresses, see below.

Meaning of Different Halt Addresses in MAT3:

MAT3/P test micrand sequences are set up to halt at different CS addresses if different conditions occur. These halt addresses are translated as follows:

Halt address = 003(HEX); Normal halt address. No error condition is indicated (displayed as 803 for the 96X mainframe).

Halt address = 847(HEX); Micrand sequence has halted because it detected a miscompare between Actual and Expected CM read data. (Micrand sequence writes the Miscompare bit of the Compare Results into UCR bit 0, then test interrupts. This causes an interrupt to CSA 47(HEX) in the event of a miscompare.)

MAT3/P TEST INITIALIZATION

The following initializations are conditionally executed after entry of test parameters and before the first test section begins executing. These initializations are performed only if the test pass count equals zero (for the first pass of the test).

- o Clear the Processor Test Mode register.
- o Master clear and clear errors in the processor.
- o Master clear and clear errors in central memory.
- o Master clear and clear errors in the IOU.
- o Initialize the Dependent Environmental Control (DEC) register of the processor:
  - Set bit 34 to enable first failure caputure.
  - Set bit 47 to disable PDMs.
  - Set bit 57 to enable Cache Set 0.
  - Set bit 61 to enable FAKECM mode.
  - Set bit 62 to force Real Memory Addressing.
  - All other bits are cleared.
- o Initialize the entire Register File (RF).
  - Write 0000 0000 0000 02D7H to RF08.
  - Write 0000 0000 95F2 C33FH to RF09.
  - Write 0000 0000 0000 022DH to RF0B.
  - Write A934 6F6B 0000 0000H to RF0C.
  - Clear all other addresses.
- o Load overlay MAT3CS0 to Control Store.
- o Verify that the microcode version date in word 0 of Control Store is compatible with the PP code version date.
- o Load overlay MAT3SM0 to Address Control (AC) Soft Control memories (type 4):
  - Addresses 40(HEX) through BF(HEX) contain a captured version of the Product Set microcode for AC Soft Control memories OUTDT1 and OUTDT2.
- o Initialize ALN Soft Control memory (type 4):
  - Address C0(HEX) is written with 0100 0100(HEX) to enable RDSB data through the Multiply/Divide Output multiplexor.
  - Addresses C1(HEX) through FF(HEX) are written with 0000 0000(HEX).

- o Initialize BDP ROM memory (type 5):
  - Addresses 000(HEX) through 5FF(HEX) are written with 0000 0000(HEX).
- o Initialize Instruction Fetch Decode RAM memory (type 6):
  - Addresses 000(HEX) through 1FF(HEX) are written with 0000 0000(HEX).
- o Initialize Reference ROM memory (REFROM) (type 3):
  - Addresses 00(HEX) through FF(HEX) contain 0000 0000(HEX).
- o Clear the central memory Bounds register.
- o Initialize the central memory DEC register:
  - Set bit 1 (disable SECDED).
  - Do not change bits 6 and 7 (timing margins).
  - Clear all other bits.
- o Clear the lowest 5 locations in central memory.
- o Start the test initialization microcode sequence (micrand sequence 0):
  - Initialize Instruction Fetch.
  - Clear the REFROM Address register, and unlock the Untranslatable Pointer (UTP).
  - Clear the 170 Error Exit Condition register.
  - Set the 180 Monitor Mode flip-flop.
  - Clear the Monitor Mask register. Clear the 170 Arithmetic flip-flop and the AS, AT, XS, XT and N registers.
  - Load a segment descriptor into Segment Map, and clear the Segment Map New-P register.
  - Clear the C170 Mode Central Memory Reference Address (RAC) register.
  - Clear the BDP internal registers and memories.
  - Set the process interval timer to FFFF FFFF(HEX), and unlock the UTP.
  - Clear the User Mask register.
  - Set the system interval timer to FFFF FFFF(HEX).
  - Clear the 170 Error Exit Condition Mask register.
  - Set the Virtual Machine Identifier register to 180 mode, and send Exchange Accept signal to IOU.
  - Clear the Debug Mask register.
  - Clear the RAC+FLC register.
  - Clear the 180 Monitor Mode flip-flop.
  - Clear the Segment Table Address register.
  - Set the 180 Monitor Mode flip-flop.
  - Clear the Page Table Address, Page Table Length, and Page Size Mask registers.
  - Clear the Segment Table Length register.
  - Purge Segment Map and clear the DAI register.
  - Purge Cache and clear the Trap Enable register to disable traps.
  - Clock Condition registers and wait two cycles.
  - Read and clear the Monitor Condition register and the User Condition register.
  - Read and clear the 170 Error Exit Condition register, and unlock the UTP.
  - Write a fake virtual instruction of 0000 0000 0000 0000(HEX) to Cache address 0 in FAKECM mode.

- Set the Return register, initialize Instruction Fetch, write zeros to the P register, and exit. Fake instruction at Cache Address 0 vectors to CSA 100(HEX), which contains a Return microand; this flushes out the pipe.
- Initialize Instruction Fetch, clock Condition register, and wait two cycles.
- Read and clear the Monitor Condition register and the User Condition register.
- Read and clear the 170 Error Exit Condition register, and unlock the UTP.
- Issue a BDP No-Op, test interrupts (to clock zeros into the Rank 50 P register), and Go to Halt at CSA 3.

o Wait for microcode to halt. If it does not halt at Control Store Address 3 within the time allotted to it, report an External Fault with error code = 6060.

o Change the contents of the processor Dependent Environmental Control (DEC) register:
  - Clear bit 57 to disable Cache.
  - Clear bit 61 to clear FAKECM mode.

o Start microand sequence 7 to read the free running counter:
  - Read the contents of the free running counter into RNG-BIAS (RF1B) in the register file, and Go to Halt at CSA 3.

o Form the random number offset value, and store into RNG-BIAS (RF1B) in the register file as follows:

  If PARAM12 and PARAM13 are both equal to FFFF(HEX), copy bytes 4-7 of the free running counter (as saved in RNG-BIAS, RF1B) into bytes 0-3 and 4-7.

  If either PARAM12 or PARAM13 is not equal to FFFF(HEX), copy contents of PARAM12 into bytes 0-1 and 4-5, and contents of PARAM13 into bytes 2-3 and 6-7.

o Read the Memory Element ID register to determine the model number of the memory to be tested.

o Read the Memory Options Installed register to determine the size of physical memory present in the memory to be tested.

o Master clear and clear errors in the processor.

SECTION DESCRIPTIONS

SECTION 0 - TEST DATA RETENTION ABILITY WITH SUPPRESSION OF REFRESH-RELATED FAULTS

This section consists of a single subsection.

Section 0, Subsection 0

This subsection tests data retention ability of each array pak (or pair of paks) in turn, with suppression of refresh-related faults.

Method:

This subsection tests all of testable memory (that block of memory delineated by PARAM10 and PARAM11) for short-term data retention capability. Detection of refresh-related failures is suppressed by performing the write/read sequence within the time period in which a memory refresh becomes necessary.

Detection of addressing-related faults is suppressed by writing the current data pattern into all of testable memory before executing any write/read/verify sequences. In the event that an addressing-related fault causes more than one memory location to output at the same time, this assures that both locations will contain the expected data.

Since each iteration tests a different memory array pak (or pair of paks), a failure in this subsection can be isolated directly from the iteration number. As an isolation aid, the location of the memory array pak currently being tested is reported in condition 0.

Microands Used:

Number	Description
1.0	Clear the MCR and UCR.
1.1	Copy MIN-WRMA (RF10) to CUR-WRMA (RF11).
1.2	Word Store RMA with: CW address from CUR-WRMA (RF11). Write data from PATTERN (RF15). Perform comparison of CUR-WRMA (RF11) with MAX-WRMA (RF12).
1.3	If CUR-WRMA (RF11) is greater than or equal to MAX-WRMA (RF12), go to 1.5.
1.4	Add 8 to CUR-WRMA (RF11), and go to 1.2.
1.5	Go to Halt at 3.
2.0	Clear the MCR and UCR.
2.1	Copy MIN-WRMA (RF10) to CUR-WRMA (RF11).
2.2	Copy PATTERN (RF15) to EXPECT (RF14).

Microcrands Used:

Number	Description
2.3	Word Store RMA with: CM address from CUR-WRMA (RF11). Write data from PATTERN (RF15).
2.4	Word Load RMA with: CM address from CUR-WRMA (RF11). Read data to ACTUAL (RF13). Perform comparison of ACTUAL (RF13) with EXPECT (RF14).
2.5	Perform comparison of CUR-WRMA (RF11) with MAX-WRMA (RF12).
2.6	If CUR-WRMA (RF11) is greater than or equal to MAX-WRMA (RF12), go to 2.8.
2.7	Add 8 to CUR-WRMA (RF11), and go to 2.3.
2.8	Go to Halt at 3.

(Microcrand sequence 1(HEX) starts at CSA 401(HEX).)
(Microcrand sequence 2(HEX) starts at CSA 402(HEX).)

Conditions:

C.0 a. Compute the MIN-WRMA (RF10) and MAX-WRMA (RF12) values for this iteration, and store in the register file.
b. Report the location of the memory array pak corresponding to the current iteration number in bytes 0 through 1; informational.

C.1 a. Write 0000 0000 0000 0000(HEX) into PATTERN (RF15) in the register file.
b. Master clear and clear errors in memory.
c. Start microcrand sequence 1; write PATTERN (RF15) to all of testable memory on the array pak(s) for this iteration.
d. Read, format, and report Status Word in Actual Results bytes 4 through 7. Report CUR-WRMA (RF11) in bytes 0 through 3. Condition fails if Status Word does not equal 0000 0803(HEX).

C.2 a. Start microcrand sequence 2; write/read/verify PATTERN (RF15) to all of testable memory on the array pak(s) being tested this iteration.
b. Read, format, and report Status Word in Actual Results bytes 4 through 7. Report CUR-WRMA (RF11) in bytes 0 through 3. Condition fails if Status Word does not equal 0000 0803(HEX).

C.3 a. Read and report UCEL1 contents; condition fails if not zero.

C.4 a. Read and report UCEL2 contents; condition fails if not zero.

C.5 a. Read contents of ACTUAL (RF13) from register file, and report in Actual Results.
b. Rewrite contents back into register file to clear possible parity error.
c. Condition fails if Actual Results do not equal 0000 0000 0000 0000(HEX).

Conditions:

C.6 a. Write FFFF FFFF FFFF FFFF(HEX) into PATTERN (RF15) in the register file.
b. Master clear and clear errors in memory.
c. Start microcrand sequence 1; write PATTERN (RF15) to all of testable memory on the array pak(s) for this iteration.
d. Read, format, and report Status Word in Actual Results bytes 4 through 7. Report CUR-WRMA (RF11) in bytes 0 through 3; condition fails if Status Word does not equal 0000 0803(HEX).

C.7 a. Start microcrand sequence 2; write/read/verify PATTERN (RF15) to all of testable memory on the array pak(s) being tested in this iteration.
b. Read, format, and report Status Word in Actual Results bytes 4 through 7. Report CUR-WRMA (RF11) in bytes 0 through 3; condition fails if Status Word does not equal 0000 0803(HEX).

C.8 a. Read and report UCEL1 contents; condition fails if not zero.

C.9 a. Read and report UCEL2 contents; condition fails if not zero.

C.10 a. Read contents of ACTUAL (RF13) from register file, and report in Actual Results.
b. Rewrite contents back into register file to clear possible parity error.
c. Condition fails if Actual Results do not equal FFFF FFFF FFFF FFFF(HEX).

C.11 a. Write 0101 0101 0101 0101(HEX) into PATTERN (RF15) in the register file.
b. Master clear and clear errors in memory.
c. Start microcrand sequence 1; write PATTERN (RF15) to all of testable memory on the array pak(s) for this iteration.
d. Read, format, and report Status Word in Actual Results bytes 4 through 7. Report CUR-WRMA (RF11) in bytes 0 through 3; condition fails if Status Word does not equal 0000 0803(HEX).

C.12 a. Start microcrand sequence 2; write/read/verify PATTERN (RF15) to all of testable memory on the array pak(s) being tested in this iteration.
b. Read, format, and report Status Word in Actual Results bytes 4 through 7. Report CUR-WRMA (RF11) in bytes 0 through 3; condition fails if Status Word does not equal 0000 0803(HEX).

C.13 a. Read and report UCEL1 contents; condition fails if not zero.

C.14 a. Read and report UCEL2 contents; condition fails if not zero.

C.15 a. Read contents of ACTUAL (RF13) from register file, and report in Actual Results.
b. Rewrite contents back into register file to clear possible parity error.
c. Condition fails if Actual Results do not equal 0101 0101 0101 0101(HEX).

End of iteration loop. If Repeat Iteration is set, repeat from C.0 with same iteration number. Otherwise, loop from C.0 for next array pak.

Exit subsection.

NOTE

The number of iterations in this subsection depends upon the size of testable memory, as specified by PARAM10 and PARAM11. Each iteration tests an array pak.

SECTION 1 - TEST DATA RETENTION ABILITY WITHOUT SUPPRESSION OF REFRESH-RELATED FAULTS

This section consists of a single subsection.

Section 1, Subsection 0

This subsection tests data retention ability of each array pak in turn, without suppression of refresh-related faults.

Method:

This subsection tests all of testable memory (that block of memory delineated by PARAM10 and PARAM11) for long-term data retention capability. Enough time is inserted between the writes and reads of each cell so that a refresh-related fault will be detected.

As in section 0, detection of addressing-related faults is supressed by writing all of testable memory with the same data pattern before any read is performed.

Since each iteration tests a different memory array pak, a failure in this subsection can be isolated directly from the iteration number. As an isolation aid, the location of the memory array pak currently being tested are repeated in condition 0.

Microands Used:

Number	Description
3.0	Clear the MCR and UCR.
3.1	Copy MIN-WRMA (RF10) to CUR-WRMA (RF11).
3.2	Copy PATTERN (RF15) to EXPECT (RF14).
3.3	Word Load RMA with: CM address from CUR-WRMA (RF11). Read data into ACTUAL (RF13). Perform comparison of ACTUAL (RF13) with EXPECT (RF14).
3.4	Perform comparison of CUR-WRMA (RF11) with MAX-WRMA (RF12).
3.5	If CUR-WRMA (RF11) is greater than or equal to MAX-WRMA (RF12), go to 3.7.
3.6	Add 8 to CUR-WRMA (RF11), and go to 3.3.
3.7	Go to Halt at 3.
1(HEX)	Refer to the description in section 0, subsection 0.

(Microand sequence 3(HEX) starts at CSA 403(HEX).)

Conditions:

C.0 a. Compute the MIN-WRMA (RF10) and MAX-WRMA (RF12) values for this iteration, and store in the register file.
b. Report the location(s) of the memory array pak(s) corresponding to the current iteration number in bytes 0 through 1; informational.

C.1 a. Write 0000 0000 0000 0000(HEX) into PATTERN (RF15) in the register file.
b. Master clear and clear errors in memory.
c. Start micrand sequence 1; write PATTERN (RF15) to all of testable memory on the array pak(s) for this iteration.
d. Read, format, and report Status Word in Actual Results bytes 4 through 7. Report CUR-WRMA (RF11) in bytes 0 through 3; condition fails if Status Word does not equal 0000 0803(HEX).

C.2 a. Start micrand sequence 3; read/verify PATTERN (RF15) to all of testable memory on the array pak(s) being tested in this iteration.
b. Read, format, and report Status Word in Actual Results bytes 4 through 7. Report CUR-WRMA (RF11) in bytes 0 through 3; condition fails if Status Word does not equal 0000 0803(HEX).

C.3 a. Read and report UCEL1 contents; condition fails if not zero.

C.4 a. Read and report UCEL2 contents; condition fails if not zero.

C.5 a. Read contents of ACTUAL (RF13) from register file, and report in Actual Results.
b. Rewrite contents back into register file to clear possible parity error.
c. Condition fails if Actual Results do not equal 0000 0000 0000 0000(HEX).

C.6 a. Write FFFF FFFF FFFF FFFF(HEX) into PATTERN (RF15) in the register file.
b. Master clear and clear errors in memory.
c. Start micrand sequence 1; write PATTERN (RF15) to all of testable memory on the array pak(s) for this iteration.
d. Read, format, and report Status Word in Actual Results bytes 4 through 7. Report CUR-WRMA (RF11) in bytes 0 through 3; condition fails if Status Word does not equal 0000 0803(HEX).

C.7 a. Start micrand sequence 3; read/verify PATTERN (RF15) to all of testable memory on the array pak(s) being tested in this iteration.
b. Read, format, and report Status Word in Actual Results bytes 4 through 7. Report CUR-WRMA (RF11) in bytes 0 through 3; condition fails if Status Word does not equal 0000 0803(HEX).

C.8 a. Read and report UCEL1 contents; condition fails if not zero.

C.9 a. Read and report UCEL2 contents; condition fails if not zero.

Conditions:

C.10 a. Read contents of ACTUAL (RF13) from register file, and report in Actual Results.
b. Rewrite contents back into register file to clear possible parity error.
c. Condition fails if Actual Results do not equal FFFF FFFF FFFF FFFF(HEX).

C.11 a. Write 0101 0101 0101 0101(HEX) into PATTERN (RF15) in the register file.
b. Master clear and clear errors in memory.
c. Start micrand sequence 1; write PATTERN (RF15) to all of testable memory on the array pak(s) for this iteration.
d. Read, format, and report Status Word in Actual Results bytes 4 through 7. Report CUR-WRMA (RF11) in bytes 0 through 3; condition fails if Status Word does not equal 0000 0803(HEX).

C.12 a. Start micrand sequence 3; read/verify PATTERN (RF15) to all of testable memory on the array pak(s) being tested in this iteration.
b. Read, format, and report Status Word in Actual Results bytes 4 through 7. Report CUR-WRMA (RF11) in bytes 0 through 3; condition fails if Status Word does not equal 0000 0803(HEX).

C.13 a. Read and report UCEL1 contents; condition fails if not zero.

C.14 a. Read and report UCEL2 contents; condition fails if not zero.

C.15 a. Read contents of ACTUAL (RF13) from register file, and report in Actual Results.
b. Rewrite contents back into register file to clear possible parity error.
c. Condition fails if Actual Results do not equal 0101 0101 0101 0101(HEX).

End of iteration loop. If Repeat Iteration is set, repeat from C.0 with same iteration number. Otherwise, loop from C.0 for next array pak until there are no more iterations.

Exit subsection.

NOTE

The number of iterations in this subsection depends upon the size of testable memory, as specified by PARAM10 and PARAM11. Each iteration tests an array pak.

SECTION 2 - TEST ADDRESS LINES BY TESTING FOR ADDRESS UNIQUENESS

This section consists of a single subsection.

Section 2, Subsection 0

This subsection tests address lines on each pak in turn, by testing for address uniqueness.

Method:

This subsection tests all of testable memory for address uniqueness, thus testing all address lines on the memory array paks and within the memory chips themselves.

To do this, the following scheme is used in C.1 through C.6. Each testable address on the array pak is initialized to a background pattern different from the write data pattern to be used. A CM Exchange function is then performed at the lowest testable address, to simultaneously read the contents of that address and then overwrite it with the write data pattern. The microcode verifies that the data read was equal to the background pattern. The CM address is then updated, and the Exchange/Verify sequence is performed for the next address on that pak. This continues until all testable addresses on the array pak have been tested. C.1 through C.4 test the addressing to the chips for the 64 data bits; the background pattern is equal to the complement of the write data pattern. C.5 and C.6 test the addressing to the chips for the 8 CM parity bits; the background pattern is chosen to generate different parity bits than those in the the write data pattern.

A fault in which an address line is stuck at a one or zero should cause more than one logical address to be mapped to the same physical address. This would be detected when the Exchange/Verify sequence is done at the second logical address that maps to the same physical address. It would be detected as the CM Read results already equal to the data pattern.

C.7 through C.13 are a repeat of C.0 through C.6, except that the memory references are made from the maximum address to the minimum address, rather than from minimum to maximum.

Since each iteration tests a different memory array pak (or pair of paks), a failure in this subsection can be isolated directly from the iteration number. As an isolation aid, the location of the memory array pak currently being tested are reported in condition 0.

Micrands Used:

Number	Description
4.0	Clear the MCR and UCR.
4.1	Copy MIN-WRMA (RF10) to CUR-WRMA (RF11).
4.2	CM Exchange RMA with: Address from CUR-WRMA (RF11). Write data from PATTERN (RF15).
4.3	Wait for second response with: Read data to ACTUAL (RF13). Perform comparison of ACTUAL (RF13) with EXPECT (RF14).
4.4	Perform comparison of CUR-WRMA (RF11) with MAX-WRMA (RF12).
4.5	If CUR-WRMA (RF11) was greater than or equal to MAX-WRMA (RF12), go to 4.7.
4.6	Add 8 to CUR-WRMA (RF11), and go to 4.2.
4.7	Go to Halt at 3.
5.0	Clear the MCR and UCR.
5.1	Copy MAX-WRMA (RF12) to CUR-WRMA (RF11).
5.2	CM Exchange RMA with: Address from CUR-WRMA (RF11). Write data from PATTERN (RF15).
5.3	Wait for second response with: Read data to ACTUAL (RF13). Perform comparison of ACTUAL (RF13) with EXPECT (RF14).
5.4	Subtract 8 from CUR-WRMA (RF11), and perform comparison of CUR-WRMA (RF11) with MIN-WRMA (RF10).
5.5	If CUR-WRMA (RF11) was less than MIN-WRMA (RF10), go to 5.7.
5.6	Go to 5.2.
5.7	Go to Halt at 3.
1(HEX)	Refer to the description in section 0, subsection 0. (Micrand sequence 4(HEX) starts at CSA 404(HEX).) (Micrand sequence 5(HEX) starts at CSA 405(HEX).)

Conditions:

C.0	a.	Compute the MIN-WRMA (RF10) and MAX-WRMA (RF12) values for this iteration, and store in the register file.
	b.	Report the location(s) of the memory array pak corresponding to the current iteration number in bytes 0 through 1; informational.
C.1	a.	Master clear and clear errors in memory.
	b.	Write data pattern of FFFF FFFF FFFF FFFF(HEX) to PATTERN (RF15).
	c.	Start micrand sequence 1(HEX); write FFs to all of testable memory.
	d.	Write data pattern of 0000 0000 0000 0000(HEX) to PATTERN (RF15).
	e.	Write FFFF FFFF FFFF FFFF(HEX) to EXPECT (RF14).
	f.	Start micrand sequence 4(HEX); exchange/verify 00s from MIN-WRMA (RF10) to MAX-WRMA (RF12).
	g.	Read, format, and report Status Word in Actual Results bytes 4 through 7. Report CUR-WRMA (RF11) in bytes 0 through 3; condition fails if Status Word does not equal 0000 0803(HEX).

Conditions:

C.2 a. Read contents of ACTUAL (RF13) from register file, and report in Actual Results.
b. Rewrite contents back into register file to clear possible parity error.
c. Condition fails if Actual Results do not equal FFFF FFFF FFFF FFFF(HEX).

C.3 a. If Repeat Condition is not set, and if no condition failed, go to c.
b. Start microrand sequence 1(HEX); write 00s to all of testable memory.
c. Write data pattern of FFFF FFFF FFFF FFFF(HEX) to PATTERN (RF15).
d. Write 0000 0000 0000 0000(HEX) to EXPECT (RF14).
e. Start microrand sequence 4(HEX); exchange/verify FFs from MIN-WRMA (RF10) to MAX-WRMA (RF12).
f. Read, format, and report Status Word in Actual Results bytes 4 through 7. Report CUR-WRMA (RF11) in bytes 0 through 3. Condition fails if Status Word does not equal 0000 0803(HEX).

C.4 a. Read contents of ACTUAL (RF13) from register file, and report in Actual Results.
b. Rewrite contents back into register file to clear possible parity error.
c. Condition fails if Actual Results do not equal 0000 0000 0000 0000(HEX).

C.5 a. If Repeat Condition is not set, and if no condition failed, go to c.
b. Start microrand sequence 1(HEX); write FF's to all of testable memory.
c. Write data pattern of 0101 0101 0101 0101(HEX) to PATTERN (RF15).
d. Write FFFF FFFF FFFF FFFF(HEX) to EXPECT (RF14).
e. Start microrand sequence 4(HEX); exchange/verify 01s from MIN-WRMA (RF10) to MAX-WRMA (RF12).
f. Read, format, and report Status Word in Actual Results bytes 4 through 7. Report CUR-WRMA (RF11) in bytes 0 through 3; condition fails if Status Word does not equal 0000 0803(HEX).

C.6 a. Read contents of ACTUAL (RF13) from register file, and report in Actual Results.
b. Rewrite contents back into register file to clear possible parity error.
c. Condition fails if Actual Results do not equal FFFF FFFF FFFF FFFF(HEX).

C.7 a. Read and report UCEL2 contents; condition fails if contents not zero.

C.8 a. Master clear and clear errors in memory.
b. Write data pattern of FFFF FFFF FFFF FFFF(HEX) to PATTERN (RF15).
c. Start microrand sequence 1(HEX); write FFs to all of testable memory.
d. Write data pattern of 0000 0000 0000 0000(HEX) to PATTERN (RF15).
e. Start microrand sequence 5(HEX); exchange/verify 00s from MAX-WRMA (RF12) to MIN-WRMA (RF10).
f. Read, format, and report Status Word in Actual Results bytes 4 through 7. Report CUR-WRMA (RF11) in bytes 0 through 3; condition fails if Status Word does not equal 0000 0803(HEX).

Conditions:

C.9 a. Read contents of ACTUAL (RF13) from register file, and report in Actual Results.
b. Rewrite contents back to register file to clear possible parity error.
c. Condition fails if Actual Results do not equal FFFF FFFF FFFF FFFF(HEX).

C.10 a. If Repeat Condition is not set, and if no condition failed, go to c.
b. Start microrand sequence 1(HEX); write 00s to all of testable memory.
c. Write data pattern of FFFF FFFF FFFF FFFF(HEX) to PATTERN (RF15).
d. Write 0000 0000 0000 0000(HEX) to EXPECT (RF14).
e. Start microrand sequence 5(HEX); exchange/verify FFs from MAX-WRMA (RF12) to MIN-WRMA (RF10).
f. Read, format, and report Status Word in Actual Results bytes 4 through 7. Report CUR-WRMA (RF11) in bytes 0 through 3; condition fails if Status Word does not equal 0000 0803(HEX).

C.11 a. Read contents of ACTUAL (RF13) from register file, and report in Actual Results.
b. Rewrite contents back to register file to clear possible parity error.
c. Condition fails if Actual Results do not equal 0000 0000 0000 0000(HEX).

C.12 a. If Repeat Condition is not set, and if no condition failed, go to c.
b. Start microrand sequence 1(HEX); write FFs to all of testable memory.
c. Write data pattern of 0101 0101 0101 0101(HEX) to PATTERN (RF15).
d. Write FFFF FFFF FFFF FFFF(HEX) to EXPECT (RF14).
e. Start microrand sequence 5(HEX); exchange/verify 01s from MAX-WRMA (RF12) to MIN-WRMA (RF10).
f. Read, format, and report Status Word in Actual Results bytes 4 through 7. Report CUR-WRMA (RF11) in bytes 0 through 3; condition fails if Status Word does not equal 0000 0803(HEX).

C.13 a. Read contents of ACTUAL (RF13) from the register file, and report in Actual Results.
b. Rewrite contents back to the register file to clear possible parity error.
c. Condition fails if Actual Results do not equal FFFF FFFF FFFF FFFF(HEX).

C.14 a. Read and report UCEL2 contents; condition fails if contents not zero.

End of iteration loop. If Repeat Iteration is set, repeat from C.0 with same iteration number. Otherwise, loop from C.0 for next array pak until there are no more iterations.

Exit subsection.

NOTE

The number of iterations in this subsection depends upon the size of testable memory, as specified by PARAM10 and PARAM11. Each iteration tests an array pak.

SECTION 3 - TEST WITH MULTIPLE ACCESSES TO EACH ADDRESS, WITH RANDOM DATA

This section consists of a single subsection.

Section 3, Subsection 0

This subsection tests each array pak (or pair of paks) in turn, with multiple accesses to each testable address and with random data.

Method:

Each memory array pak (CYBER 960 and 962 memory) is tested in turn, one iteration per pak. Each testable address on the pak is accessed four times in succession. They are accessed via a write, which is followed by two exchanges and a read. For each address, these four CM functions are issued by the processor as fast as they can be accepted by the memory. Each of the three words of write data (one for the write and one for each of the two exchanges) contains different random data. After all four memory accesses have been made to one address, the read results are verified and the next address is tested.

Since each iteration tests a different memory array pak (or pair of paks), a failure in this subsection can be isolated directly from the iteration number. As an isolation aid, the location of the memory array pak currently being tested are reported in condition 1.

Micrands Used:

Number	Description
6.0	Clear the MCR and UCR.
6.1	Copy MIN-WRMA (RF10) to CUR-WRMA (RF11).
6.2	Clear Xs, set Xt = 1.
6.3	Perform generation of next random data word into RF address (30(HEX) + Xs).
6.4	Copy contents of RF address (30(HEX) + Xs) into RF address (28(HEX) + Xs).
6.5	Write zeros to RF address (38(HEX) + Xs), and if Xs is greater than Xt, go to 6.7.
6.6	Increment Xs, and go to 6.3.
6.7	Word Store RMA with: CM address from CUR-WRMA (RF11). Write data from RF30.
6.8	CM Exchange RMA with: CM address from CUR-WRMA (RF11). Write data from RF31.
6.9	Wait for second response with: Read data to RF38.
6.10	CM Exchange RMA with: CM address from CUR-WRMA (RF11). Write data from RF32.

Microands Used:

Number	Description
6.11	Wait for second response with: Read data to RF39.
6.12	Word Load RMA with: CM address from CUR-WRMA (RF11). Read data to RF3A.
6.13	Clear Xs, set Xt = 1, and perform one no-op.
6.14	ALN performs a 64-bit compare of the contents of RF address (28(HEX) + Xs) with contents of RF address (38(HEX) + Xs). Write encoded results into bits 32 and 33 of TEMP (RF18).
6.15	Shift TEMP (RF18) left 33 bits, and write shifted results to the UCR.
6.16	Test interrupts, and if Xs is greater than Xt, go to 6.18.
6.17	Increment Xs, and go to 6.14.
6.18	Perform comparison of CUR-WRMA (RF11) with MAX-WRMA (RF12).
6.19	If CUR-WRMA (RF11) was greater than or equal to MAX-WRMA (RF12), go to 6.21.
6.20	Add 8 to CUR-WRMA (RF11), and go to 6.2.
6.21	Go to Halt at 3.

(Microand sequence 6(HEX) starts at CSA 406(HEX).)

Conditions:

C.0 a. Report the 64-bit bias values of the random number generators. These were formed from the contents of the free running counter during test initialization; informational.

C.1 a. Read, save, and report the current state of the random data generator; informational.

C.2 a. Compute the MIN-WRMA (RF10) and MAX-WRMA (RF12) values for this iteration, and store in the register file.
b. Report the location of the memory array pak corresponding to the current iteration number in bytes 0 through 1; informational.

C.3 a. Restore the state of the random data generator to the state it was in when captured in C.1.
b. Master clear and clear errors in memory.
c. Start microand sequence 6(HEX). Execute multiple accesses with random data to/from each testable address on array pak(s) currently being tested in this iteration.
d. Read, format, and report Status Word in Actual Results bytes 4 through 7. Report CUR-WRMA (RF11) in bytes 0 through 3; condition fails if Status Word does not equal 0000 0803(HEX).

C.4 a. Read and report UCEL1 contents; condition fails if not zero.

Conditions:

C.5 a. Read and report UCEL2 contents; condition fails if not zero.

C.6 a. Read results of first CM Exchange from RF38, and report in Actual Results.
b. Rewrite data into RF38 to clear any parity errors.
c. Read write data for Word Store microand from RF28, and report as Expected Results.
d. Condition fails if Actual Results do not equal Expected Results.

C.7 a. Read results of first CM Exchange from RF39, and report in Actual Results.
b. Rewrite data into RF39 to clear any parity errors.
c. Read write data for Word Store microand from RF29, and report as Expected Results.
d. Condition fails if Actual Results do not equal Expected Results.

C.8 a. Read results of first CM Exchange from RF3A, and report in Actual Results.
b. Rewrite data into RF3A to clear any parity errors.
c. Read write data for Word Store microand from RF2A, and report as Expected Results.
d. Condition fails if Actual Results do not equal Expected Results.

End of iteration loop. If Repeat Iteration is set, repeat from C.2 with same iteration number. Otherwise, loop from C.1 for next array pak until there are no more iterations.

Exit subsection.

NOTE

The number of iterations in this subsection depends upon the size of testable memory, as specified by PARAM10 and PARAM11. Each iteration tests an array pak (or pair of paks).

SECTION 4 - TEST WITH RANDOM DATA

This section consists of a single subsection.

Section 4, Subsection 0

This subsection tests each array pak (or pair of paks) in turn, with random data.

Method:

Each memory array pak (CYBER 960 and 962 memory) is tested in turn, one iteration per pak. Each address on the pak within the area of testable memory is first written full of random data. A CM Exchange is then performed to each address, writing new random data and reading and verifying the old random data. Finally, each address is then read and verified.

In each of the three steps (CM Write, CM Exchange, and CM Read), the CM requests are issued in blocks of eight words, as fast as they can be accepted by memory. The address, write data, and expected read data for each of the eight CM requests have been previously calculated and stored in the register file. Similarly, the read results for each CM request are saved in the register file until all eight requests have executed; only then are the read results verified.

Since each iteration tests a different memory array pak, a failure in this subsection can be isolated directly from the iteration number. As an isolation aid, the location of the memory array pak currently being tested are reported in condition 2.

Micrands Used:

Number	Description
A.0	Clear the MCR and UCR.
A.1	Copy MIN-WRMA (RF10) to CUR-WRMA (RF11).
A.2	Copy CUR-WRMA (RF11) to BLOCK-FWA (RF1A), clear Xs, set Xt = 6, and perform one no-op.
A.3	Copy CUR-WRMA (RF11) into RF address (20(HEX) + Xs), and perform generation of next word of write data into RF address (30(HEX) + Xs).
A.4	If Xs is greater than Xt, go to A.6.
A.5	Add 8 to CUR-WRMA (RF11), increment Xs, and go to A.3.
A.6	Copy BLOCK-FWA (RF1A) to CUR-WRMA (RF11).
A.7	Word Store RMA with: CM address from RF20. Write data from RF30.
A.8	Word Store RMA with: CM address from RF21. Write data from RF31.

Micrands Used:

Number	Description
A.9	Word Store RMA with: CM address from RF22. Write data from RF32.
A.10	Word Store RMA with: CM address from RF23. Write data from RF33.
A.11	Word Store RMA with: CM address from RF24. Write data from RF34.
A.12	Word Store RMA with: CM address from RF25. Write data from RF35.
A.13	Word Store RMA with: CM address from RF26. Write data from RF36.
A.14	Word Store RMA with: CM address from RF27. Write data from RF37.
A.15	Add 40(HEX) to CUR-WRMA (RF11), and perform comparison of CUR-WRMA (RF11) with MAX-WRMA (RF12).
A.16	If CUR-WRMA (RF11) was equal to or greater than MAX-WRMA (RF12), go to A.18.
A.17	Go to A.2.
A.18	Go to Halt at 3.
B.0	Clear the MCR and UCR.
B.1	Copy MIN-WRMA (RF10) to CUR-WRMA (RF11).
B.2	Copy CUR-WRMA (RF11) to BLOCK-FWA (RF1A), clear Xs, set Xt = 6, and perform one no-op.
B.3	Clear RF address (38(HEX) + Xs), and perform generation of next word of write data into RF address (30(HEX) + Xs).
B.4	Perform prediction of next word of expected read data into RF address (2816 + Xs).
B.5	Copy CUR-WRMA (RF11) into RF address (20(HEX) + Xs), and if Xs is greater than Xt, go to B.7.
B.6	Add 8 to CUR-WRMA (RF11), increment Xs, and go to B.3.
B.7	Copy BLOCK-FWA (RF1A) to CUR-WRMA (RF11).
B.8	CM Exchange with: CM address from RF20. Write data from RF30.
B.9	Wait for second response with: Read data to RF38.
B.10	CM Exchange with: CM address from RF21. Write data from RF31.
B.11	Wait for second response with: Read data to RF39.
B.12	CM Exchange with: CM address from RF22. Write data from RF32.

Microrands Used:

Number	Description
B.13	Wait for second response with: Read data to RF3A.
B.14	CM Exchange with: CM address from RF23. Write data from RF33.
B.15	Wait for second response with: Read data to RF3B.
B.16	CM Exchange with: CM address from RF24. Write data from RF34.
B.17	Wait for second response with: Read data to RF3C.
B.18	CM Exchange with: CM address from RF25. Write data from RF35.
B.19	Wait for second response with: Read data to RF3D.
B.20	CM Exchange with: CM address from RF26. Write data from RF36.
B.21	Wait for second response with: Read data to RF3E.
B.22	CM Exchange with: CM address from RF27. Write data from RF37.
B.23	Wait for second response with: Read data to RF3F.
B.24	Clear Xs, set Xt = 6, and perform one no-op.
B.25	Copy RF address (28(HEX) + Xs) to EXPECT (RF14).
B.26	Copy RF address (38(HEX) + Xs) to ACTUAL (RF13), and perform comparison of ACTUAL (RF13) with EXPECT (RF14).
B.27	If Xs is greater than Xt, go to B.29.
B.28	Add 8 to CUR-WRMA (RF11), increment Xs, and go to B.25.
B.29	Add 8 to CUR-WRMA (RF11), and perform comparison of CUR-WRMA (RF11) with MAX-WRMA (RF12).
B.30	If CUR-WRMA (RF11) was equal to or greater than MAX-WRMA (RF12), go to B.32.
B.31	Go to B.2.
B.32	Go to Halt at 3.
C.0	Clear the MCR and UCR.
C.1	Copy MIN-WRMA (RF10) to CUR-WRMA (RF11).
C.2	Copy CUR-WRMA (RF11) to BLOCK-FWA (RF1A), clear Xs, set Xt = 6, and perform one no-op.
C.3	Copy CUR-WRMA (RF11) into RF address (20(HEX) + Xs), and perform prediction of next word of expected read data into RF address (28(HEX) + Xs).
C.4	Clear RF address (38(HEX) + Xs), and if Xs is greater than Xt, go to C.6
C.5	Add 8 to CUR-WRMA (RF11), increment Xs, and go to C.3.
C.6	Copy BLOCK-FWA (RF1A) to CUR-WRMA (RF11).

Microrands Used:

Number	Description
C.7	Word Load RMA with: CM address from RF20. Read data to RF30.
C.8	Word Load RMA with: CM address from RF21. Read data to RF31.
C.9	Word Load RMA with: CM address from RF22. Read data to RF32.
C.10	Word Load RMA with: CM address from RF23. Read data to RF33.
C.11	Word Load RMA with: CM address from RF24. Read data to RF34.
C.12	Word Load RMA with: CM address from RF25. Read data to RF35.
C.13	Word Load RMA with: CM address from RF26. Read data to RF36.
C.14	Word Load RMA with: CM address from RF27. Read data to RF37.
C.15	Clear Xs, set Xt = 6, and perform one no-op.
C.16	Copy RF address (28(HEX) + Xs) to EXPECT (RF14).
C.17	Copy RF address (38(HEX) + Xs) to ACTUAL (RF13), and perform comparison of ACTUAL (RF13) with EXPECT (RF14).
C.18	If Xs is greater than Xt, go to C.20.
C.19	Add 8 to CUR-WRMA (RF11), increment Xs, and go to C.(HEX).
C.20	Add 8 to CUR-WRMA (RF11), and perform comparison of CUR-WRMA (RF11) with MAX-WRMA (RF12).
C.21	If CUR-WRMA (RF11) was equal to or greater than MAX-WRMA (RF12), go to C.23.
C.22	Go to C.2.
C.23	Go to Halt at 3.
	(Microrand sequence A(HEX) starts at CSA 40A(HEX).) (Microrand sequence B(HEX) starts at CSA 40B(HEX).) (Microrand sequence C(HEX) starts at CSA 40C(HEX).)

Conditions:

C.0 a. Report the 64-bit bias values of the random number generators. These were formed from the contents of the free running counter during test initialization; informational.

Conditions:

C.1 a. Read, save, and report the current state of the random data generator; informational.

C.2 a. Report the location(s) of the memory array pak(s) corresponding to the current iteration in bytes 0 through 1 (and bytes 2 through 3); informational.
b. Compute the MIN-WRMA (RF10) and MAX-WRMA (RF12) values for this iteration, and store in the register file.

C.3 a. Restore the random data generator to the state it was in when it was captured in C.1.
b. Master clear and clear errors in memory.
c. Start microrand sequence A(HEX); write random data to each testable address in the array pak(s) being tested in this iteration.
d. Read, format, and report Status Word in Actual Results bytes 4 through 7. Report CUR-WRMA (RF11) in bytes 0 through 3; condition fails if Status Word does not equal 0000 0803(HEX).

C.4 a. Read and report UCEL1 contents; condition fails if not zero.

C.5 a. Read and report UCEL2 contents; condition fails if not zero.

C.6 a. Read, save, and report the current state of the random data generator; informational.

C.7 a. Restore the random data generator to the state it was in when it was captured in C.6.
b. Set the random data predictor to the state the random data generator was in when captured in C.1.
c. Master clear and clear errors in memory.
d. Start microrand sequence B(HEX); exchange and verify random data to/from each testable address in the array pak(s) being tested in this iteration.
e. Read, format, and report Status Word in Actual Results bytes 4 through 7. Report CUR-WRMA (RF11) in bytes 0 through 3; condition fails if Status Word does not equal 0000 0803(HEX).

C.8 a. Read and report UCEL1 contents; condition fails if not zero.

C.9 a. Read and report UCEL2 contents; condition fails if not zero.

C.10 a. Read contents of ACTUAL (RF13), and report in Actual Results.
b. Read contents of EXPECT (RF14), and report in Expected Results.
c. Condition fails if Actual Results do not equal Expected Results.

C.11 a. Set the random data predictor to the state the random data generator was in when captured in C.6.
b. Master clear and clear errors in memory.
c. Start microrand sequence C(HEX); read and verify random data from each testable address in the array pak(s) being tested in this iteration.
d. Read, format, and report Status Word in Actual Results bytes 4 through 7. Report CUR-WRMA (RF11) in bytes 0 through 3; condition fails if Status Word does not equal 0000 0803(HEX).

Conditions:

C.12 a. Read and report UCEL1 contents; condition fails if not zero.

C.13 a. Read and report UCEL2 contents; condition fails if not zero.

C.14 a. Read contents of ACTUAL (RF13), and report in Actual Results.
b. Read contents of EXPECT (RF14), and report in Expected Results.
c. Condition fails if Actual Results do not equal Expected Results.

End of iteration loop. If Repeat Iteration is set, repeat from C.2 with same iteration number. Otherwise, loop from C.1 for next array pak until there are no more iterations.

Exit subsection.

NOTE

The number of iterations in this subsection depends upon the size of testable memory, as specified by PARAM10 and PARAM11. Each iteration tests an array pak.

SECTION 5 - TEST ALL OF TESTABLE MEMORY WITH RANDOM DATA AND RANDOM ADDRESSES

This section consists of a single subsection.

Section 5, Subsection 0

This subsection tests all of testable memory with random data and random addresses.

Method:

This subsection first writes random data into each location of testable memory. These locations are written in random order using a random address generator, which assures that each location is written only once.

Each address in testable memory is then read in the same random order in which it was written, and the read results are verified.

Memory references are done in blocks of eight words, with three microrands per memory reference. The address, write data, and/or expected read data for each of the eight memory requests have been previously calculated and stored in the register file. Similarly, the read results for each memory request are saved in the register file until all eight requests have executed; only then are the read results verified.

Microrands Used:

Number	Description
8.0	Clear the MCR and UCR.
8.1	Copy MIN-WRMA (RF10) to ITERATION (RF16), and perform initialization of the random address generator.
8.2	Clear Xs and set Xt = 6.
8.3	Perform generation of next random address into CUR-WRMA (RF11).
8.4	Copy CUR-WRMA (RF11) into RF address (20(HEX) + Xs), and perform generation of next random data word into RF address (30(HEX) + Xs).
8.5	If Xs is greater than Xt, go to 8.7.
8.6	Increment Xs and go to 8.3.
8.7	Word Store RMA with: CM address from RF20. Write data from RF30.
8.8	Word Store RMA with: CM address from RF21. Write data from RF31.
8.9	Word Store RMA with: CM address from RF22. Write data from RF32.

Microrands Used:

Number	Description
8.10	Word Store RMA with: CM address from RF23. Write data from RF33.
8.11	Word Store RMA with: CM address from RF24. Write data from RF34.
8.12	Word Store RMA with: CM address from RF25. Write data from RF35.
8.13	Word Store RMA with: CM address from RF26. Write data from RF36.
8.14	Word Store RMA with: CM address from RF27. Write data from RF37.
8.15	Add eight to ITERATION (RF16), and perform comparison of ITERATION (RF16) with MAX-WRMA (RF12).
8.16	If ITERATION (RF16) was greater than or equal to MAX-WRMA (RF12), go to 8.18.
8.17	Go to 8.2.
8.18	Go to Halt at 3.
9.0	Clear the MCR and UCR.
9.1	Copy MIN-WRMA (RF10) to ITERATION (RF16), and perform initialization of the random address generator.
9.2	Clear Xs, and set Xt = 6.
9.3	Perform generation of next random address into CUR-WRMA (RF11).
9.4	Copy CUR-WRMA (RF11) into RF address (20(HEX) + Xs), and perform prediction of next random data word into RF address (28(HEX) + Xs).
9.5	Write zeros to RF address (38(HEX) + Xs), and if Xs is greater than Xt, go to 9.7.
9.6	Increment Xs and go to 9.3.
9.7	Word Load RMA with: CM address from RF20. Read data to RF38.
9.8	Word Load RMA with: CM address from RF21. Read data to RF39.
9.9	Word Load RMA with: CM address from RF22. Read data to RF3A.
9.10	Word Load RMA with: CM address from RF23. Read data to RF3B.
9.11	Word Load RMA with: CM address from RF24. Read data to RF3C.
9.12	Word Load RMA with: CM address from RF25. Read data to RF3D.

Microrands Used:

Number	Description
9.13	Word Load RMA with: CM address from RF26. Read data to RF3E.
9.14	Word Load RMA with: CM address from RF27. Read data to RF3F.
9.15	Write zeros to INDEX (RF19), clear Xs and N, set Xt = 6, and perform one no-op.
9.16	ALN performs a 64-bit compare on the contents of RF address (28(HEX) + Xs) with contents of RF address (38(HEX) + Xs). Write encoded compare results into bits 32 and 33 of TEMP (RF18).
9.17	Shift TEMP (RF18) left 33 bits, and write shifted results to the UCR.
9.18	Test interrupts, and if Xs is greater than Xt, go to 9.20.
9.19	Add 1 to INDEX (RF19), increment N and Xs, and go to 9.(HEX).
9.20	Add 8 to ITERATION (RF16), and perform comparison of ITERATION (RF16) with MAX-WRMA (RF12).
9.21	If ITERATION (RF16) was greater than or equal to MAX-WRMA (RF12), go to 9.23.
9.22	Go to 9.2.
9.23	Go to Halt at 3.

(Microrand sequence 8(HEX) starts at CSA 408(HEX).)
(Microrand sequence 9(HEX) starts at CSA 409(HEX).)

Conditions:

C.0 a. Report the 64-bit bias values for the random number generators. These were formed from the contents of the free running counter during test initialization; informational.

C.1 a. Read, save, and report the current state of the random address generator; informational.

C.2 a. Read, save, and report the current state of the random data generator; informational.

Conditions:

C.3 a. Compute the MIN-WRMA (RF10) and MAX-WRMA (RF12) values to test all of testable memory (per PARAM10 and PARAM11), and write these values to the register file.
b. Restore the random address generator to the state it was in when it was captured in C.1.
c. Restore the state of the random data generator to the state it was in when it was captured in C.2.
d. Master clear and clear errors in memory.
e. Start microrand sequence 8(HEX); write random data to random addresses throughout all of testable memory.
f. Read, format, and report Status Word in Actual Results bytes 4 through 7. Report address of last CM write in bytes 0 through 3. Last CM address is contained in RF27; condition fails if Status Word does not equal 0000 0803(HEX).

C.4 a. Read and report UCEL1 contents; condition fails if not zero.

C.5 a. Read and report UCEL2 contents; condition fails if not zero.

C.6 a. Restore the random address generator to the state it was in when it was captured in C.1.
b. Set the random data predictor to the state the random data generator was in when it was captured in C.2.
c. Master clear and clear errors in memory.
d. Start microrand sequence 9(HEX); read and verify random data from random addresses throughout all of testable memory.
e. Read, format, and report Status Word in Actual Results bytes 4 through 7. Report address of last CM read in bytes 0 through 3. Last CM address is contained in RF at address [20(HEX) + INDEX (RF19)]; condition fails if Status Word does not equal 0000 0803(HEX).

C.7 a. Read UCEL1 contents, and report in Actual Results; condition fails if not zero.

C.8 a. Read UCEL2 contents, and report in Actual Results; condition fails if not zero.

C.9 a. Read from the register file the read results from the last CM read, and report in Actual Results. Last CM read results are found at RF address [38(HEX) + INDEX (RF19)].
b. Rewrite the CM read results back into the register file to clear possible parity errors.
c. Read the expected read results for the last CM exchange from the register file at address [28(HEX) + INDEX (RF19)], and report in Expected Results.
d. Condition fails if Actual Results do not equal Expected Results.

End of iteration loop. If Repeat Iteration is set, repeat from C.3 with same iteration number. Otherwise, loop from C.1 until there are no more iterations.

Exit subsection.

NOTE

The number of iterations in this subsection is determined from the contents of PARAM9.

THIS PAGE HAS INTENTIONALLY BEEN LEFT BLANK

Abs

Absolute.

AC or A/C

Address Control.

ACMIC

Address Control Functional Microrand.

ADATB/RDSB/B

Refers to B Operand data in Multiply/ Divide unit test MDT3/P.

ADATC/RDSC/C

Refers to C Operand data in Multiply/ Divide unit test MDT3/P.

Adder

Refers to Large Adder in ALN test ANT3/P. Refer to Multiply Final Adder in test MDT3/P.

Adder Bit 47 FF

Large Adder Result bit 47 flip-flop in ALN test ANT3/P.

Adrs

Address.

ALN

Arithmetic and Logical Network.

ALU

Arithmetic and Logical Unit.

AOR

Address-Out-of-Range.

Arith

Arithmetic.

ASCII

American Standard Code for Informa- tion Interchange.

ASE

Address Specification Error.

ASID

Active Segment Identifier.

Assy

Assembly.

Aux

Auxiliary.

B

Refers to the operand input to ALN on the RSDB data path. Used in ALN test ANT3/P.

BCD

Binary Coded Decimal.

BD

Branch Delta.

BDP

Business Data Processor.

Bfr

Buffer.

Bin

Binary.

Bkpt

Breakpoint.

1BN

Byte Number.

Bound

Boundary.

Br

Branch.

C

Refers to the operand input to the ALN on the RDSC data path. Used in ALN test ANT3/P.

CA

Condition Action.

CABR

Cache Address Buffer Register.

CAE

Condition Action Extension.

CAR

Cache Address Register.

CBP

Code Base Pointer.

CCR

Clock Condition Register.

CEL

Corrected Error Log.

C.GT.B

C Greater than B signal from the SKG in ALN test ANT3/P.

Chan

Channel.

Clk

Clock.

Clr

Clear.

CM

Central Memory.

CMC

Central Memory Control.

Cnt

Count.

Cntr

Counter.

Coef

Coefficient.

Com

Common.

Comp

Complement.

Cond

Condition.

Confl

Conflict.

Cont

Control.

Conv

Convert, Converter.

Cor

Corrected.

CPU

Central Processing Unit.

1CS

Condition Sensing.

CSA

Control Store Address.

CST

Control Store.

CW1

Control Word 1.

CW2

Control Word 2.

DAI

Data Interchange.

DCDR

Decoder.

DCI

Data control information. Two bytes of information that give the maintenance register address.

DEC

Dependent Environment Control, Decimal.

Decr

Decrement.

Descr

Descriptor.

Destn

Destination.

Disassy

Disassembly.

Div

Divide.

Divisor Rgtr

Divide network register internal to LSI CI divide arrays.

Dlyd

Delayed.

DMR

Debug Mask Register.

Dsbl

Disable.

DSS

Data Subfunction Select Microrand Field.

EBCDIC

Extended Binary Coded Decimal Inter- change Code.

ECC

Error Correction Code.

ECR

C170 Exit Mode Condition Register.

EI

Environment Interface.

EMMR or EMR

Exit Mode Mask Register.

Enbl

Enable.

Encdr

Encoder.

ESE

Environment Specification Error.

EXCH

Exchange.

Exp

Exponent.

Exponent Aux

Exponent Auxiliary board in ALN.

Ext

External.

Fctn

Function.

FF

Flip-flop.

FIFO

First In/First Out.

FLC

C170 Mode Central Memory Field Length.

FNO

Fanout.

FRC

Free Running Counter.

FU

Functional Unit.

Gen

Generator.

Genl

General.

GME

Used by test CTT3/P to indicate a field (bits 62 and 63) within the general micrand.

Hldg

Holding.

i

Operand Register Designator.

IACY

C170 Parcel Select Multiplexer (IFT3/P test).

IAHY

Instruction Assembly Hybrid Multi- plexer (IFT3/P test).

IAPL

C180 Parcel Select Multiplexer (IFT3/P test).

IB02

Instruction Buffer Rank 02 (IFT3/P test).

IB12 valid

Used in ALN test ANT3/P. Instruction Buffer Rank 12 valid signal from Instruction Fetch. This goes active when IF has finished fetching the next instruction from CM.

ICC

Instruction Completion Control.

ICP

Instruction Control Pipeline.

IDS

Immediate Data Select Micrand Field.

IDX

Indexed.

IF

Instruction Fetch.

II

Instruction Issue.

Immed

Immediate.

Incr

Increment.

Indef

Indefinite.

Inf

Infinite.

Inh

Inhibit.

Inp

Input.

Instr

Instruction.

Int

Internal.

Integer Mux

Integer/Exponent Multiplexer in ALN.

Intrpt

Interrupt.

IOU

Input/Output Unit.

ISE

Instruction Specification Error.

IU

Instruction Unit.

j

Operand Register Designator.

JPS

Job Process State.

k

Operand Register Designator.

K

170 18-Bit Immediate Operand.

K/L

Key/Lock.

Kypt

Keypoint.

l

BDP Operand Length.

Ld

Load.

LM

Local Memory.

LMMIC

Local Memory Functional Microrand field.

LOS

Loss of Significance.

LRR

Used in ALN test ANT3/P and Multiply/ Divide test MDT3/P, and ICT3/P to indicate a live register read.

LRU

Least Recently Used.

LRW

Used in ALN test ANT3/P and Multiply/ Divide test MDT3/P and ICT3/P to indicate a live register write.

SD

Least Significant Digit.

LSI

Large Scale Integration.

LSM

Last Symbol Mask.

LSMPRE

LOAD/STORE Multiple Address Adder.

Lwr

Lower.

MAC

Maintenance Access Control.

Maint

Maintenance.

MAR

Micrand Address Register.

MBE

Multiple Bit Error.

MC

Master Clear.

MCH

Maintenance Channel.

MCMR

Monitor Condition Mask Register. Also refer to MMR.

MCR

Monitor Condition Register.

Mem

Memory.

Micr

Micrand.

Micrand Constant

Contents of ALN FU Micrand bits 0 through 6.

MIS

Miscellaneous Micrand Field.

Misc

Miscellaneous.

MMR

Monitor Mask Register.

MOP

Micro operation.

MPS

Monitor Process State.

MR

Maintenance Register.

MSB

Most Significant Bit.

MSC

Microand Sequence Control. A field (bits 0 through 15) within the general microand.

MSD

Most Significant Digit.

MSNZB

Most Significant Nonzero Byte.

Mult

Multiply.

Mux

Multiplexer.

Neg

Negative.

Norm

Normalize.

NPGK

New P Global Key.

NPLK

New P Local Key.

ns

Nanosecond.

o

BDP Operand Address.

Op

Operand.

Opcode

Operation Code.

OPGK

Old P Global Key.

OPLK

Old P Local Key.

OPI

Operand Issue.

Opn

Operation.

OSB

Operating System Bounds.

Out

Output.

Ovfl

Overflow.

P

Program Address.

Par

Parity.

PDM

Processor Detected Malfunction.

PE

Parity Error.

PFA

Page Frame Address.

PFS

Processor Fault Status.

Ph

Phase.

PIT

Process Interval Timer.

PMF

Performance Monitoring Facility.

PN

Page Number.

PO

Page Offset.

PONR

Point Of No Return.

Pos

Positive.

PP

Peripheral Processor.

Prev

Previous.

Pri

Priority.

Prod

Product.

PSM

Page Size Mask.

PSWF

Page Table Search Without Find.

PTA

Page Table Address.

PTD

Page Table Descriptor.

PTL

Page Table Length.

PTM

Processor Test Mode.

PVA

Process Virtual Address.

PW

Partial Write.

Q

C180 16-Bit Immediate Operand.

Quad

Quadrant.

Quotient Rgtr

Divide network register internal to the LSI CI arrays.

R/W

Read/Write.

RAC

C170 Mode Central Memory Reference Address.

RAM

Random Access Memory.

RCL

Read and Clear Lock.

Rcvr

Receiver.

RDS

Register/Data Select.

RDSA

In ALN test ANT3/P, refers to the micrand field that provides the register file address on a write of ALN data and to the data path through DAI for that data.

RDSB

In ALN test ANT3/P, refers to the following data path to ALN. The specified register file is output to the RDSB bus, transferred to the ALN on ADATB, and is passed through the multiple/divide unit and output multiplexer to the general network.

RDSC

In ALN test ANT3/P, refers to following data path. The specified register file is sent to the RDSC bus, trans- ferred to ADATC, then goes to the general network.

Ref

Reference.

Regen

Regenerator.

Rel

Relative.

Rem

Remainder.

Req

Request.

Resync

Resynchronize, Resynchronizer, Resynchronization.

RF

Register File.

RFOx

Used in ALN test ANT3/P and Multiply/ Divide test MDT3/P to identify the register file being used. x is 5, 6, 7, 8, 9, A, B, C, D, E or F (Hexadecimal).

Rgtr

Register.

RMA

Real Memory Address.

ROM

Read Only Memory.

RSL

Read and Set Lock.

RS64

Right Shift 64 signal from the SKG.

Rxx

Identifies ranks in IF and ICP in Test CTT3/P.

SBE

Single Bit Error.

SC

Soft Control.

SCM

Soft Control Memory.

SCT

Special Character Table.

SECDED

Single Error Correction/Double Error Detection.

Seg

Segment.

Sel

Select.

Sep

Separate.

Seq

Sequence.

Shift Mux

Shift Network Multiplexer.

Shifter

Shift Network.

Sig

Significant.

SIT

System Interval Timer.

SKG

Shift Count Generator.

SM

Segment Map.

SMMIC

Segment Map Functional Microrand Field.

SMO(x)

Segment Descriptor Multiplexer Output bit x.

SMPDM

Segment Map Processor Detected Malfunction.

SN

Segment Number.

Spec

Specification.

SPID

Segment/Page Identifier.

STA

Segment Table Address.

STL

Segment Table Length.

Str

Stream.

Subtr

Subtract.

Suppr

Suppression.

SV

Specification Value.

SVA

System Virtual Address.

Sw

Switch.

Syn

Syndrome.

Sync

Synchronize.

Termn

Terminate.

TFA

Tag File Address.

ubit

Microrand Bit.

UCEL1

Uncorrected Error Log 1.

UCEL2

Uncorrected Error Log 2.

UCR

User Condition Register.

UMR

User Mask Register.

Unbr

Unbranch.

Uncond

Unconditional.

Uncor

Uncorrectable.

Undfl

Underflow.

Unlog

Unlogged.

Upr

Upper.

usec

Microsecond.

UTP

Untranslatable Pointer.

Ux

Used in Multiply/Divide test MDT3/P to identify Major Cycle Control bits 0 through 14.

VMID

Virtual Machine Identifier.

Vx

Used in Multiply/Divide test MDT3/P to identify Minor Cycle Control bits 0 through 15.

DS

Write Data Select.

WOI

Word of Interest.

Xfer

Transfer.

Xltr

Translator.

Xmtr

Transmitter.

XOR

Exclusive OR.

ZIF

Zero Insertion Force.

ZF

Zero Flag.

REGISTER FILE MEMORY MAP

RF Address (Hex)	Descriptive Name	Description
00	(none)	Initialized to zero during test initialization. Written nonzero if CSA 0 is executed.
01	(none)	Live Register Write data and Live Register Read results.
02-07	(none)	Not used.
08	RAG-STATE	Current state of the random address generator.
09	RDG-STATE	Current state of the random data generator.
0A	RDP-STATE	Current state of the random data predictor.
0B	RAG-PRIME	32-bit prime used by random address generator.
0C	RDG-PRIME	32-bit prime used by random data generator and random data predictor.
0D-0F	(none)	Not used.
10	MIN-WRMA	Minimum word RMA (inclusive) to be tested.
11	CUR-WRMA	Current word RMA being tested.
12	MAX-WRMA	Maximum word RMA (inclusive) to be tested.
13	ACTUAL	Actual results.
14	EXPECT	Expected results.
15	PATTERN	Current data pattern.
16	ITERATION	Current count of CM addresses accessed.
17	ADRS-MASK	Entry parameter to random address generator routine.
18	TEMP	Temporary storage for microcode routines.
19	INDEX	Describes which word of an 8-word block of memory references is currently being executed.

RF Address (Hex)	Descriptive Name	Description
1A	BLOCK-FWA	Word RMA of first word in 8-word block of memory references.
1B	RNG-BIAS	64-bit bias added to random address or data.
1C	M	Used in random address generator routine.
1D-1F	(none)	Not used.
20-27	(none)	8-word block of CM word addresses.
28-2F	(none)	8-word block of expected CM read results.
30-37	(none)	8-word block of CM write data.
38-3F	(none)	8-word block of actual CM read results.

CONTROL STORE MEMORY MAP

The following is a Control Store memory map for the MAT3/P test. It indicates the locations of the various types of microrands used by the MAT3/P test.

Any Control Store locations not appearing in this map are not used by MAT3/P and contain all zeros. If Control Store were to branch to one of these locations, the MSC field containing all zeros would cause Control Store to branch to CSA 0, which contains a microrand that writes a nonzero value to RF address 0 and spins, not setting halt.

CS Address (Hexadecimal)	Contents
0	Contains a microrand that writes nonzero value to RF address 0, and spins without setting halt. This microrand will be executed if MAT3/P microcode branches to a CS address that contains all zeros.
3	Normal halt address. Contains a Spin-Halt microrand.
5	MCH Stop vector point. Contains a Spin-Halt microrand.
8	Unimplemented Instruction vector point. Contains a Spin-Halt microrand.
10	Unconditional Exit vector point. Contains a Spin-Halt microrand.

CS Address (Hexadecimal)	Contents
20	Conditional Exit vector point. Contains an Exit micrand.
30-32	MAC Assisted Read and Write vector points. Contain Spin-Halt micrands.
40-47	Microtrap interrupt addresses. Contain Spin-Halt micrands.
80-1FE	Entry points for miscellaneous micrand sequences and microcode subroutines.
1FF	Entry point for a 180 instruction with an FF(HEX) opcode. Contains a micrand that branches to the CS address previously loaded into the Return register.
200-27F	Entry points for micrand sequences that read contents of Live Register addresses 00 through 7F(HEX) respectively (where utilized) into RF address 01.
300-37F	Entry points for micrand sequences that write contents of RF address 01 into Live Register addresses 00 through 7F(HEX) respectively (where utilized).
380	Unexecutable Instruction vector point. Contains a Spin-Halt micrand.
400-4FF	Entry points for test micrand sequences numbered 00 through FF16 (where utilized).
500-7FF	Scratch area for use of all micrand sequences that contain more than one micrand. The first micrand of all such micrand sequences (found at its entry point in CS) branches to the location in this block of Control Store addresses at which the second and all subsequent micrands of that micrand sequence are found.

CONTROL DATA

4201 North Lexington Avenue Saint Paul, Minnesota 55126-6198